Práctica 2:

COSTE DE FUNCIONES

OBJETIVO

- Calcular costes teóricos y reales de funciones. Representarlos gráficamente.
- Introducción al manejo de GNUPLOT.

MATERIAL A ENTREGAR

- costes.cpp
- Ficheros de datos (*.dat).
- Gráficas (a.png, b.png, at.png, bt.png, medio.png)
- makefile
- Ejercicios apartado 1 (previo).

INTRODUCCIÓN A GNUPLOT

El programa GNUPLOT es una utilidad para la visualización de ficheros de datos, que utilizaremos para representar los ficheros generados por los programas de cálculo de costes que vamos a realizar. Los comandos básicos del GNUPLOT se muestran a continuación:

`plot 2*x`: Dibuja la gráfica y = 2*x.

`plot 'fichero' w p`: Dibuja la primera columna del fichero en el eje de abscisas y la segunda en el eje de ordenadas como puntos.

`plot 'fichero' w l` : Igual que antes pero une los puntos mediante líneas.

`plot 'fichero' w l, 'fichero2' w l`: Dibuja los dos ficheros en la misma gráfica. Se pueden dibujar varios ficheros sin más que seguir el mismo esquema.

`plot 'fichero' w l, funcion`: Dibuja el fichero y la función especificada.

`save 'fichero.plt'`: Guarda la ultima gráfica realizada.

`load 'fichero.plt'`: Lee fichero.plt y muestra la gráfica.

Para guardar una imagen en fichero:

`set terminal png`: Para crear una imagen en formato png. Con ‘set terminal’ podemos ver otras opciones.

`set output 'fichero.png'`: Indica el fichero de salida.

`plot sin(x)` : Guarda la grafica de sin(x) en el fichero ‘fichero.png’.

DESARROLLO

1. Cuestiones previas. Cálculo del coste teórico de funciones.

Calcular el coste exacto en pasos de las siguientes funciones. En las funciones con más de 1 caso, calcula caso mejor y caso peor. Los sumatorios deben resolverse totalmente.

```
a)
func fa(n: Nat) dev k: Nat
{   // Pre: n > 0
    k ← 0;
    para i ← 1 hasta n-1
        para j ← 1 hasta i-1
           k ← k + j
        fpara
    fpara

}

b)
func fb(v: Vector[1..n] de {0,1}) dev cic:Bool
{
    a ← 1
    b ← n
    mientras (a < b) ∧ (v[a] = v[b]) hacer
        a ← a+1
        b ← b-1
    fmientras
    si (a >= b) entonces
        cic ← cierto
    sino
        cic ← falso
    fsi
}
```

2. Representación del coste teórico de funciones

Representar el coste de las funciones calculadas en el apartado 1 mediante gnuplot. Si la función tiene más de 1 caso, representa el caso mejor y peor en la misma gráfica. Guarda las gráficas en los ficheros ‘a.png’ y ‘b.png’.

Para calcular si el coste calculado es correcto, se deberán **implementar las funciones** y añadir un contador para calcular el número de pasos realizados en cada función. El contador se pasará por referencia a las funciones.

Implementar la función **evaluar**, que permitirá volcar en un fichero de salida (file) los valores de la talla y su correspondiente coste (pares n t(n)) de una función (t) para un rango suficiente de n (entre ini y fin) con un incremento determinado (inc). El parámetro t será un puntero a función, lo que nos permitirá pasarle cualquier función que cumpla la especificación dada. La cabecera de esta función será:

```
void evaluar(unsigned ini, unsigned fin, int inc, float (*t)(unsigned), ostream
& file);
```

Para que la función de la que se quiere calcular el coste se adapte al parámetro de función definido, se deberán crear funciones auxiliares que conviertan el tamaño del problema en parámetros para las funciones, y que devuelvan el coste en pasos en lugar del resultado de la función. Por ejemplo, la función auxiliar para la función a sería la siguiente:

```
float faAux(unsigned n)
{
    pasos = 0;
    fa(n, pasos);
    return pasos;
}
```

Implementar un programa costes.cpp, que valiéndose de las funciones anteriores, genere los ficheros **at.dat**, **bt.dat**. Estos ficheros contendrán los costes teóricos en pasos de las funciones a y b, que habrá que representar posteriormente utilizando el GNUPLOT. Los rangos irán de 4 a 100 con incrementos de 4. **Representar** estos costes **junto con los calculados anteriormente** y guardarlos en los ficheros 'at.png' y 'bt.png'.

3. Cálculo del coste medio

Calcular el coste medio de la función b. Se deberá calcular **tanto de forma teórica en papel como mediante el ordenador** y se deberá compararlos para comprobar si coinciden. Para calcular el coste mediante el ordenador, se deberá definir una función que genere vectores de tamaño *n* conteniendo los valores 0 y 1 de forma aleatoria. Completar el programa costes.cpp para que también calcule estos costes.

Se deberán visualizar los costes juntos en la gráfica medio.png.